CMPSC 465 Spring 2024 | Data Structures & Algorithms | Mehrdad Mahdavi and David Koslicki

Worksheet 11

Wednesday, April 3, 2024

1. **Vertex Cover.** Given an undirected graph $G = (V,E)$, a Vertex Cover is a set of vertices $V' \subseteq V$ such that every edge $e \in E$ is incident on (touches) some vertex in V'. Give a greedy algorithm for choosing vertices such that, if the optimal solution uses k vertices, the greedy algorithm will never use more than:
   (a) $k\ln(|E|)$ vertices.
   (b) $2k$ vertices.

2. **Worst-case Greedy Set Cover Instances.**

   Recall the set cover problem:

   Input: A set of n elements B and sets $S_1,\ldots,S_m \subseteq B$.

   Output: A selection of the S_i whose union is B (i.e. that contain every element of B).

   Cost: Number of sets picked.

   It was proven in class that the greedy algorithm for the Set Cover problem returns a solution that is at most $O(\log n)$ above the optimal solution. In this problem, we will prove that it also applies for $\Omega(\log n)$, which gives $\Theta(\log n)$

   Show that for any integer n that is a power of 2, there is an instance of the set cover problem (i.e. a collection of sets $S_1,\ldots,S_m$) with the following properties:

   i. There are n elements in the base set B.
   ii. The optimal cover uses just two sets.
   iii. The greedy algorithm picks at least $\Omega(\log n)$ sets.

3. **Maximum Non-Adjacent Sum.** You are given an array A = $(a_1, a_2, ..., a_\text{n})$ with n (possibly negative) integers. Design an algorithm to find a subsequence A_1 of A such that A_1 does not contain adjacent elements of A (i.e. if $a_\text{k} \in \text{A}_1$ then $a_\text{k-1} \notin \text{A}_1$ and $a_\text{k+1} \notin \text{A}_1$) and that the sum of all integers in A_1 is maximized. Your algorithm should run in $O(n)$ time.

4. **Longest Common Subsequence.** Given two strings $x = x_1x_2\ldots x_n$ and $y = y_1y_2\ldots y_m$, we wish to find the length of their *longest common subsequence*, that is, the largest k for which there are indices $i_1 < i_2 < \cdots < i_k$ and $j_1 < j_2 < \cdots < j_k$ with $x_{i_1}x_{i_2}\ldots x_{i_k} = y_{j_1}y_{j_2}\ldots y_{j_k}$, show how to do this in time $O(mn)$. Note that in general, a "subsequence" may not be contiguous, but a "substring" must be contiguous.

5. **File Reconstruction.** You are given a string of n characters $s[1,\ldots,n]$, which you believe to be a corrupted text document in which all punctuation has vanished (so that it looks something like "itwasthebestoftimes..."). You wish to reconstruct the document using a dictionary, which is available in the form of a Boolean function dict($\cdot$): for any string w,

$$\texttt{dict}(w) = \begin{cases} \texttt{true} & \text{if } w \text{ is a valid word} \\ \texttt{false} & \text{Otherwise} \end{cases} \tag{1}$$

(a) Give a dynamic programming algorithm that determines whether the string $s[.]$ can be reconstituted as a sequence of valid words. The running time should be at most $O(n^2)$, assuming calls to dict take unit time.

(b) In the event that the string is valid, make your algorithm output the corresponding sequence of words